Лабораторна робота №6

Регресія. Прогноз ціни Ethereum

Мета роботи: набути практичних навичок роботи у побудові моделі для прогнозування цін на валютному ринку блокчейну

Література

tf.keras.callbacks.EarlyStopping: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

sklearn.metrics.mean_absolute_error: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html

Зміст роботи

Завдання 1. Побудувати модель для прогнозу ціни криптовалюти ефір(ETH).

Оригінальна концепція Ethereum була представлена у 2013 році Віталіком Бутеріним під час випуску документу про Ethereum, а в 2015 році платформу Ethereum запустили Бутерін і Джозеф Лубін разом із кількома іншими співзасновниками. Ethereum позиціонує себе як електронну програмовану мережу, яку кожен може побудувати для запуску криптовалют і децентралізованих програм. На відміну від біткойну, максимальний тираж якого становить 21 мільйон монет, кількість ETH, яку можна створити, необмежена, хоча час, необхідний для обробки блоку ETH, обмежує кількість ефіру. Ще одна відмінність між Ethereum і Bitcoin полягає в тому, як мережі обробляють комісію за обробку транзакцій. Ці збори відомі як «газ» у мережі Ethereum і сплачуються учасниками транзакцій Ethereum.

[https://www.blockchain.com/explorer/assets/eth]

Набори даних:

Для роботи використано три набора даних blockchain_data.csv - дані були зібрані з крипто-валютної біржи Blockchain, historical_data.csv – дані були зібрані з крипто-валютної біржs Binance, twitter_sentiments.csv- наведено сентиментальний аналіз твітів стосовно валюти(позитивні, негативні, нейтральні)

- Завдання рекомендується виконувати в Google Colaboratory.
- Бібліотеки, які знадобляться для роботи:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout, LSTM
from sklearn.metrics import mean_squared_error
from datetime import datetime
```

- Завантажити файли (використати назви дата фреймів prices_df, blockchain_df, twitter_sentiments).

– Продивитися перші n рядків файлів.

– Створити функції для макетів графіків:

```
def line_plot(train, test, label1 = None, label2 = None, title='', lw=2):
    fig, ax = plt.subplots(1, figsize=(13, 7))
    if 'time' in train.keys():
        ax.plot(train['time'].apply(lambda x: datetime.fromtimestamp(x)),
train['close'], label=label1, linewidth=lw)
        ax.plot(test['time'].apply(lambda x: datetime.fromtimestamp(x)), t
est['close'], label=label2, linewidth=lw)
    else:
        ax.plot(train['close'], label = label1, linewidth = lw)
        ax.plot(test['close'], label = label2, linewidth = lw)
    ax.set_ylabel('Ціна [USD]', fontsize = 14)
    ax.set_title(title, fontsize = 16)
    ax.legend(loc='best', fontsize = 16)
    plt.show()

def plot_result(actual, predicted):
    fig, ax = plt.subplots(1, figsize=(13, 7))
    ax.plot(actual, label = 'actual', linewidth = 1)
    for key in predicted.keys():
        ax.plot(predicted[key], label = key, linewidth = 1)
    ax.set_ylabel('Ціна [USD]', fontsize = 14)
    ax.legend(loc='best', fontsize = 16)
    plt.show()
```

– Об’єднати набори даних:

```
merged_df = pd.merge(pd.merge(blockchain_df, prices_df, left_on = 'time',
right_on = 'close_time'), twitter_sentiments, on = 'time')
```

– Видалити непотрібні данні. Відсортувати за часом.

– Продивитися перші n рядків отриманого набору даних.

Підготувати дані для машинного навчання.

– Підготувати функції: для розбиття набору даних на тренувальні та тестові дані; для нормалізації даних; для вилучення даних; prepare_data() – використовується для завантаження підготовлених даних. Завантаження та збереження даних за допомогою кількох процесів (distributed settings) призведе до пошкодження даних.

```
def train_test_split(df, test_size = 0.2):
    split_row = len(df) - int(test_size * len(df))
    train_data = df.iloc[:split_row]
    test_data = df.iloc[split_row:]
    return train_data, test_data

def normalise_zero_base(df):
    coefs = df.iloc[0].values
```

```
    for i in range(len(coefs)):
        if coefs[i] == 0:
            coefs[i] = 1

    return df / coefs - 1

def extract_window_data(df, window_len):
    window_data = []
    for idx in range(len(df) - window_len):
        data = normalise_zero_base(df[idx: (idx + window_len)].copy())
        window_data.append(data.values)
    return np.array(window_data)

def prepare_data(df, window_len, test_size = 0.2):
    target_col = 'close'

    train_data, test_data = train_test_split(df, test_size = test_size)
    X_train = extract_window_data(train_data, window_len)
    X_test = extract_window_data(test_data, window_len)
    y_train = train_data[target_col][window_len:].values
    y_test = test_data[target_col][window_len:].values
    y_train = y_train / train_data[target_col][:-window_len].values - 1
    y_test = y_test / test_data[target_col][:-window_len].values - 1

    return train_data, test_data, X_train, X_test, y_train, y_test
```

Тестувати побудованої моделі завжди потрібно на даних, які не брали участь у навчанні. Такі дані називають тестовими. Дані, що беруть участь у навчанні, називаються навчальними. Розбиття набору даних на навчальні та тестові дані можна за допомогою функції train_test_split.

– Виконати розбиття набору даних на навчальні та тестові дані. Застосуємо цю функцію до даних, до тестової частини відносимо 20%, до навчальної — решта 80%. Оскільки були використані стандартні параметри, вихідні дані спочатку були перемішані, а потім розділені.

– Отримати інформацію про кількість строк і рядків в навчальній і тестову вибірки.

– Побудувати графік відповідності ціна - дата тренувального і тестового наборів даних.

```
line_plot(train, test,'тренувальний набір','тестувальний набір', title='')
```

Почати навчання

– Припинити навчання, коли показник, що відстежується, перестав покращуватися:

```
early_stopping_callback = tf.keras.callbacks.EarlyStopping(patience = 10,
restore_best_weights = True)
```

```
def lr_scheduler(epoch, lr):
    if epoch > 0 and epoch % 10 == 0:
        return lr * 0.75
    return lr

lr_scheduler_callback = tf.keras.callbacks.LearningRateScheduler(lr_schedu
ler)
```

– Функція для побудувати LSTM моделі. Нейронна мережа складається з рівня LSTM, за яким слідує рівень 20% Dropout і щільний шар із лінійною функцією активації.

```
def build_lstm_model(input_data, neurons, dropout, optimizer):
    model = Sequential()
    model.add(LSTM(neurons, input_shape = (input_data.shape[1], input_data
.shape[2])))
    model.add(Dropout(dropout))
    model.add(Dense(units = 1))
    model.add(Activation('linear'))

    model.compile(loss = 'mse', optimizer = optimizer)
    return model
```

– Задати параметри.

```
window_len = 10
neurons = 256
epochs = 10
batch_size = 32
dropout = 0.2
optimizer = 'adam'

df = merged_df[['close']]
```

– Навчити модель.

```
train, test, X_train, X_test, y_train, y_test = prepare_data(df, window_le
n)
model = build_lstm_model(X_train, neurons, dropout, optimizer)
history = model.fit(np.asarray(X_train).astype('float32'), np.asarray(y_tr
ain).astype('float32'), epochs = epochs, batch_size = batch_size, shuffle
= True, callbacks = [early_stopping_callback, lr_scheduler_callback])
```

– Використати MSE. Mean Squared Error вимірює кількість помилок у статистичних моделях. Зведення різниць у квадрат усуває негативні значення і гарантує, що середня квадратична помилка завжди буде більшою або дорівнює нулю. Тільки ідеальна модель без помилок дає нульовий MSE. А на практиці такого не буває. Якщо взяти квадратний корінь із MSE, отримаємо середню квадратичну помилку (RMSE), яка використовує

натуральні одиниці вимірювання. Іншими словами, MSE аналогічно дисперсії, тоді як RMSE схоже на стандартне відхилення.

```
targets = test['close'][window_len:]
preds = model.predict(X_test).squeeze()
print(' MSE:', mean_squared_error(preds, y_test))
preds = test['close'].values[window_len:] * (preds + 1)
preds = pd.Series(index = targets.index, data = preds)
plot_result(targets.values, {'price': preds.values})
```

Як показник оцінки також можна використати середню абсолютну похибку (Mean Absolute Error (MAE)) Це середнє за тестовою вибіркою абсолютних відмінностей між фактичними та прогнозованими спостереженнями.

– Використати MAE. Порівняти результати.

Тестування

– Обрати змінні з кожної підгрупи :

```
data_props = {
    'price': ['close'],
    'historical': ['close', 'open', 'low', 'high', 'volume', 'trades_amoun
t'],
    'blockchain': ['max_value', 'average_amount', 'average_fee', 'average_
gas_price', 'gas_used', 'transaction_amount'],
    'twitter': ['positive', 'neutral']
}
```

– Згрупувати дата сет відповідностей:

```
data_sets = [['historical'], ['historical', 'blockchain'], ['historical',
'twitter'], ['historical', 'blockchain', 'twitter']]
```

– Провести навчання на кожному угрупуванні:

```
targets = test['close'][window_len:]
results = {}

mse_results = {}

for item in data_sets[:4]:
    fields = []
    for field_type in item:
        fields.extend(data_props[field_type])

    df = merged_df[fields]
    train, test, X_train, X_test, y_train, y_test = prepare_data(df, windo
w_len)
    current_model = build_lstm_model(X_train, neurons, dropout, optimizer)
```

```
    current_model.fit(np.asarray(X_train).astype('float32'), np.asarray(y_
train).astype('float32'), epochs = epochs, batch_size = batch_size, shuffl
e = True, callbacks = [early_stopping_callback, lr_scheduler_callback])

    preds = current_model.predict(X_test).squeeze()
    key = ', '.join(item)
    mse_results[key] = mean_squared_error(preds, y_test)
    preds = test['close'][window_len:] * (preds + 1)
    preds = pd.Series(index = targets.index, data = preds)
    results[key] = preds.values
```

– Побудувати графік отриманого результату

```
plot_result(targets.values, results)
```

– Вивести показники оцінки по кожному угрупуванню:

```
rating = sorted(mse_results.items(), key = lambda x: x[1])
for item in rating:
    print(item[0], '-', item[1])
```

– Зробити висновки

Завдання 2. Зробити прогноз на майбутні періоди.

Завдання 3. Побудувати модель для прогнозу ціни криптовалюти bitcoin (https://www.kaggle.com/datasets/mczielinski/bitcoin-historical-data).

Контрольні запитання

1. За допомогою якої функції можна розбити набір даних на навчальні та тестові дані ?
2. Для чого використовується LSTM?
3. Як працює LSTM?
4. Які існують методи виміру точності моделі?
5. Що таке середня абсолютна похибка?
6. Що таке середня квадратична похибка?
7. За якою формулою можна знайти MAE?
8. За якою формулою можна знайти MSE?
9. Які відмінності між MAE і MSE?
10. Як ви інтерпретуєте середню абсолютну похибку?
11. Як обчислити середню абсолютну похибку в Python?
12. Як обчислити середню квадратичну помилку за допомогою Scikit-Learn?